ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
ПРАВОСЛАВНЫЙ СВЯТО-ТИХОНОВСКИЙ ГУМАНИТАРНЫЙ УНИВЕРСИТЕТ
(ПСТГУ)

Факультет информатики и прикладной математики
Кафедра информатики

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ ПО ДИСЦИПЛИНЕ

«Операционные системы и оболочки»

02.03.03 «Математическое обеспечение и администрирование информационных систем»

Профиль:
Администрирование информационных систем

Квалификация выпускника: бакалавр

Форма обучения: очная

Москва, 2019 г.

Год начала обучения по учебному плану: 2019

Фонд оценочных средств для текущего контроля успеваемости разработан на основе рабочей программы дисциплины «Операционные системы и оболочки», входящей в состав образовательной программы 02.03.03 «Математическое обеспечение и администрирование информационных систем».

Для текущего контроля успеваемости студентов в целях проверки процесса достижения результатов обучения и уровня сформированности компетенций, проводятся тесты, и 10 практических работ.

За все задания начисляются баллы. Всего может быть начислено за работу в семестре до 70 баллов. Из них:

Тесты – до 20 баллов

Лабораторные работы – до 50 баллов (по 5 баллов за каждую работу)

Примерные вопросы тестовых заданий

Вопрос 1. Что произойдет при выполнении команды cd без параметров?
а) на экране высветится имя текущей директории
б) текущей директорией станет домашняя директория пользователя
в) текущей директорией станет корневая директория

Вопрос 2. Что произойдет в результате выполнения команды
`cp -r aaa bbb`, где `aaa` и `bbb` - имена существующих директорий (директория `bbb` – пустая, все необходимые права доступа имеются)?
а) команда выдаст сообщение об ошибке
б) все файлы из директории aaa рекурсивно скопируются под своими именами в директорию `bbb`
в) в директорию `bbb` рекурсивно скопируется директория `aaa` под своим именем

Вопрос 3. Что возвращает системный вызов getuid()?
а) имя пользователя, запустившего программу
б) идентификатор пользователя, запустившего программу
в) идентификатор пользователя, создавшего исполняемый файл

Вопрос 4. Относительное имя файла определяется
а) текущей директорией процесса
б) домашней директорией пользователя
в) типом файла

Вопрос 5. В текущей директории находятся регулярные файлы с именами .a, .ab, b, ac, bdd, cdd.c и пустая директория aaac — все с правами доступа rwx. Какие файлы останутся в директории после выполнения команды rm *[b-d]
а) .a
б) .ab
в) b
г) ac
д) bdd
е) cdd.c
ж) aaac

Вопрос 6. Какие из перечисленных ниже редакторов файлов являются экранными редакторами:
vi
ed
joe

Вопрос 7. Кто может изменить у файла идентификатор его хозяина?
системный администратор
текущий хозяин файла
пользователь из текущей группы хозяев файла

Вопрос 8. В директории с правами для доступа некоторого пользователя wx находится исполняемый файл с правами для доступа этого пользователя rx. Что может пользователь сделать с файлом?
узнать имя этого файла
выполнить файл
удалить файл

Вопрос 9. Какие из перечисленных ниже функций и системных вызовов ввода-вывода поддерживают потоковую передачу данных:
fgets() read() fread() printf() write()

Вопрос 10. Время жизни средства связи FIFO в вычислительной системе определяется:
временем жизни взаимодействующих процессов
временем жизни операционной системы
временем жизни информации на жестком диске

Вопрос 11. Какой тип связи обеспечивает FIFO:
симплексную связь полудуплексную связь дуплексную связь

Информация, хранящаяся в FIFO, располагается: в адресном пространстве пользовательского процесса в адресном пространстве ядра операционной системы на жестком диске

Вопрос 12. Если два процесса не имеют общего прародителя, создавшего потоковое средство связи, то чем из перечисленного ниже они могут воспользоваться для взаимодействия?
только pipe
только FIFO
ни тем, ни другим

Вопрос 13. Если мы требуем, чтобы файл на диске отсутствовал и был создан в момент открытия, то какую комбинацию флагов для вызова open() можно применить:
O_RDWR | O_CREAT O_RDONLY O_WRONLY | O_CREAT | O_EXCL

Вопрос 14. Информация, хранящаяся в pipe, располагается: в адресном пространстве пользовательского процесса в адресном пространстве ядра операционной системы на жестком диске

Вопрос 15. Какая комбинация флагов в системном вызове open() не позволит процессу заблокироваться при открытии FIFO: O_RDONLY O_RDONLY | O_NDELAY O_RDWR

Вопрос 16. Время жизни средств связи System V IPC (если их специально не удалять) в вычислительной системе определяется:
временем жизни взаимодействующих процессов
временем жизни операционной системы
временем жизни информации на жестком диске

Вопрос 17. Сколько нитей исполнения может быть ассоциировано с одной и той же функцией в одном процессе?
а) не более одной
б) одна
в) произвольное количество

Вопрос 18. Какие из комбинаций специальных значений для флагов и ключа в системном вызове shmget() являются допустимыми (т. е. не приведут к ошибке):
а) IPC_CREAT и IPC_EXCL
б) IPC_PRIVATE и IPC_EXCL
в) IPC_PRIVATE и IPC_CREAT

Вопрос 19. Какие из перечисленных средств связи, которые использует процесс, могут остаться доступными без специальных системных вызовов (pipe(), open(), shmget()) после успешного выполнения системного вызова exec():
а) pipe FIFO
б) разделяемая память
в) System V IPC

Вопрос 20. Какие данные из информации, сообщаемой командой ipcs shm, требуются в качестве параметра команде ipcrm:
а) идентификатор пользователя, создавшего сегмент разделяемой памяти
б) размер сегмента разделяемой памяти
в) IPC дескриптор сегмента разделяемой памяти

Вопрос 21. Какие переменные являются разделяемыми для нескольких нитей исполнения одного процесса: глобальные статические переменные, т. е. статические переменные, описанные вне функций в языке C локальные статические переменные, т. е. статические переменные, описанные внутри функций в языке C локальные динамические переменные

Вопрос 22. Какие значения, возвращаемые функцией pthread_create(), свидетельствуют о возникновении ошибочной ситуации? значения > 0 значение 0 значения < 0

Вопрос 23. Через разделяемую память могут взаимодействовать: только процесс-ребенок и его родитель, создавший разделяемую память близкородственные процессы, имеющие общего прародителя, создавшего разделяемую память произвольные процессы в системе

Вопрос 24. Для совместной работы массив семафоров могут использовать: только процесс-ребенок и его родитель, создавший массив семафоров близкородственные процессы, имеющие общего прародителя, создавшего массив семафоров произвольные процессы в системе

Вопрос 25. Какая из операций над семафорами SYSTEM V IPC является аналогом операции V(S) над семафорами Дейкстры: A(S,n) D(S,n) Z(S) не имеет аналогов

Вопрос 26. Сразу после создания массива из трех семафоров с идентификатором IPC semid процесс выполняет следующие действия:

```
{
struct sembuf mybuf[2];
mybuf[0].sem_op = 2;
mybuf[0].sem_flg = 0;
mybuf[0].sem_num = 0;
mybuf[1].sem_op = 0;
mybuf[1].sem_flg = 0;
mybuf[1].sem_num = 1;
semop(semid, &mybuf, 2);
}
```

Чему будут равны значения семафоров с номерами 0 и 1 после их выполнения, если другие процессы в системе доступа к ним не имеют?
Варианты ответа:
а) 2 и 0
б) 0 и 2
в) процесс не вернется из вызова semop

Вопрос 27. Семафоры System V IPC по сравнению с семафорами Дейкстры являются:
а) более мощным средством синхронизации (все, что можно реализовать семафорами Дейкстры, можно реализовать с их помощью, а обратное не является верным)
б) эквивалентными по возможностям
в) менее мощным средством синхронизации (все, что можно реализовать с их помощью, можно реализовать семафорами Дейкстры, а обратное не является верным)

Вопрос 29. Некоторый процесс, создавший массив семафоров, успешно выполнил системный вызов exec(). Будет ли доступен массив семафоров в новом пользовательском контексте:
нет
да, без дополнительного запроса информации от других процессов или операционной системы
да, после дополнительного запроса информации от других процессов или операционной системы

Вопрос 30. Если не предпринимать специальных действий по удалению созданной очереди сообщений, то ее время жизни будет определяться:
временем жизни взаимодействующих процессов
временем жизни операционной системы
временем жизни вычислительного комплекса

Вопрос 31. Два процесса собираются обмениваться сообщениями через единственную очередь. Могут ли они создать ее и получить соответствующий дескриптор, воспользовавшись оба вызовами msgget() с ключом IPC_PRIVATE и флагами 0666 | IPC_CREAT?
да, всегда
нет, никогда
да, при некоторых дополнительных условиях

Вопрос 32. В очереди сообщений находится 6 сообщений S1, S2, S3, S4, S5, S6 с соответствующими типами 2, 1, 3, 5, 1, 2. Некоторый процесс в цикле выполняет системный вызов msgrcv с четвертым параметром, равным 3. Сколько сообщений и в каком порядке он

прочитает до своего блокирования? 5 сообщений: S2, S5, S1, S6, S3 6 сообщений: S1, S2, S3, S4, S5, S6 3 сообщения: S1, S2, S3

Вопрос 33. Какая длина должна быть указана в качестве третьего параметра системного вызова msgrcv()?:
максимальная длина полезной части информации в принимаемом сообщении (т.е. длина сообщения без его типа)
максимальная полная длина принимаемого сообщения
максимальная длина области памяти, доступной процессу, начиная с адреса, заданного вторым параметром

Вопрос 34. В очереди сообщений находится 6 сообщений S1, S2, S3, S4, S5, S6 с соответствующими типами 2, 1, 3, 5, 1, 2. Некоторый процесс в цикле выполняет системный вызов msgrcv с четвертым параметром, равным 0. Сколько сообщений и в каком порядке он прочитает до своего блокирования?
ни одного
6 сообщений: S1, S2, S3, S4, S5, S6
6 сообщений: S2, S5, S1, S6, S3, S4

Вопрос 35. Какая длина должна быть указана в качестве третьего параметра системного вызова msgsnd()?:

полная длина полезной части информации в сообщении (т.е. длина сообщения без его типа)
полная длина передаваемого сообщения

полная длина области памяти, доступной процессу, начиная с адреса, заданного вторым параметром

Вопрос 36. Сколько различных типов файлов существует в операционной системе Linux?

а) 4;

б) 6;

в) 8.

Вопрос 37. Некоторый процесс выполняет системный вызов unlink() для файла, у которого счетчик числа жестких связей равен 1. Когда файл будет удален с диска? непосредственно после выполнения вызова unlink() после завершения работы процесса после того, как счетчик числа открытий в системной таблице открытых файлов станет равным 0

Вопрос 38. При отображении файла в память процесс использовал в системном вызове mmap() флаг MAP_PRIVATE. Будут ли изменения в образе файла, лежащего в памяти, отображены на дисковое пространство? да, всегда да, если процесс использует системный вызов munmap() нет, не будут

Вопрос 39. Какие из следующих типов файлов в системе Linux могут находиться в нетерминальных узлах графа файловой системы? регулярные файлы файлы типа «связь» директории

Вопрос 40. К каким из перечисленных ниже типов файлов можно организовать жесткую связь в операционной системе Linux? регулярные файлы файлы типа «связь» файлы типа «устройство» директории

Вопрос 41. Что полностью и однозначно характеризует файл, хранящийся в файловой системе UNIX на конкретном устройстве? номер его индексного узла номер первого блока, содержащего данные файла полное имя файла

Вопрос 42. После открытия регулярного файла один процесс порождает другой. Через некоторое время процесс-родитель читает из этого файла 20 байт, а затем процесс-ребенок, не открывая файл заново, читает из него же 20 байт. Что можно сказать о прочитанной информации?

а) это будет одна и та же информация

б) 20 байт, прочитанных «ребенком», лежат в файле сразу за 20-ю байтами, прочитанными «родителем»

в) ничего сказать нельзя, все зависит от предыстории поведения «ребенка» и «родителя»

Вопрос 43. Какие из следующих типов файлов в системе Linux могут находиться в терминальных узлах графа файловой системы? регулярные файлы файлы типа «связь» файлы типа «устройство» директории

Вопрос 44. К каким из перечисленных ниже типов файлов можно организовать жесткую связь в операционной системе Linux? регулярные файлы файлы типа «связь» файлы типа «устройство» директории

Вопрос 45. Создание потоковых клиента и сервера для стека UNIX Domain протоколов
По аналогии с программами в предыдущем примере модифицируйте тексты программ TCP клиента и сервера для сервиса echo (программа15–16-3.c и программа15–16-4.c) для потокового общения в семействе UNIX Domain протоколов. Откомпилируйте их и убедитесь в правильном функционировании.

Вопрос 46. Какой из уровней семейства TCP/IP отвечает за доставку информации от сокета отправителя к сокету получателя? уровень сетевого интерфейса уровень Internet транспортный уровень

Вопрос 47. Какие из перечисленных ниже системных вызовов используются в стандартной схеме общения для TCP-клиента?
accept()
connect()
bind()

Вопрос 48. В каком из системных вызовов в структуре, описывающей полный адрес сокета, указатель на которую является параметром системного вызова, и при какой ситуации для семейства протоколов TCP/IP в качестве номера порта может быть задано значение 0?
bind() в UDP-сервере для сокета, предназначенного для приема первоначального запроса от клиента
bind() в UDP-сервере для сокета, предназначенного только для отправки информации
connect()

Вопрос 49. В каком из системных вызовов в структуре, описывающей полный адрес сокета, указатель на которую является параметром системного вызова, и при какой ситуации для семейства протоколов TCP/IP в качестве номера порта может быть задано значение 0?
bind() в UDP-клиенте
bind() в TCP-сервере
sendto()

Вопрос 50. В каком из системных вызовов в структуре, описывающей полный адрес сокета, указатель на которую является параметром системного вызова, для семейства протоколов TCP/IP в качестве IP-адреса может быть задано значение INADDR_ANY?
bind()
sendto()
connect()

Вопрос 51. Какой из уровней семейства TCP/IP отвечает за доставку информации от компьютера-отправителя к компьютеру-получателю в масштабах всей сети?
уровень сетевого интерфейса
уровень Internet
транспортный уровень

Вопрос 52. Какие из перечисленных ниже системных вызовов используются в стандартной схеме общения для UDP-сервера?
а) accept()
б) connect()
в) bind()

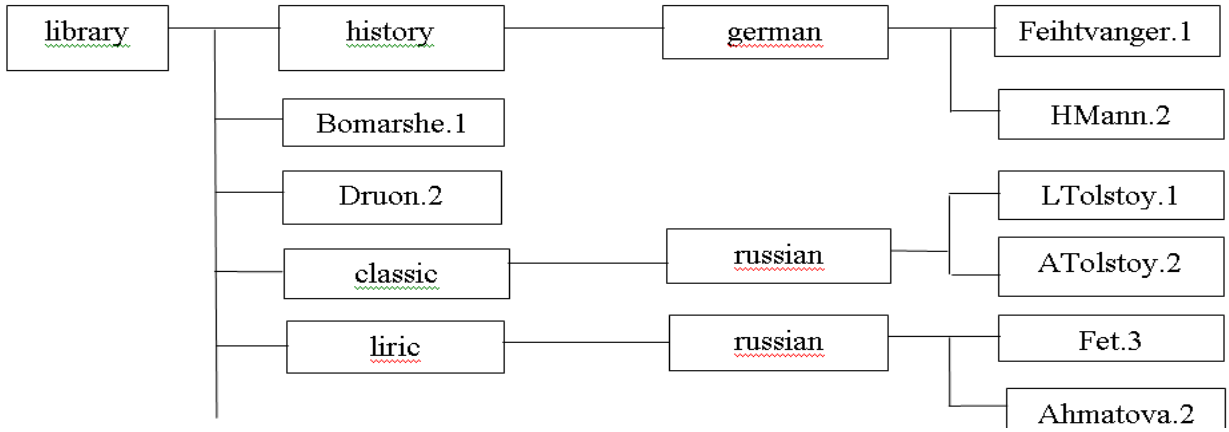
Вопрос 53. Какой из уровней семейства TCP/IP отвечает за доставку информации от сокета отправителя к сокету получателя?
а) уровень сетевого интерфейса
б) уровень Internet
в) транспортный уровень

Критерии оценивания тестовых заданий – пусть q - доля правильных ответов (вычисляется делением количества данных верных ответов на общее число вопросов теста), тогда количество начисляемых баллов равно $[q \cdot 20]$ где $[x]$ – целая часть x.
Балльные оценки приводятся к традиционным следующим образом: отлично = 18-20 баллов, хорошо – 15-17 баллов, удовлетворительно – 12-14 баллов, неудовлетворительно – менее 12 баллов. Студент, сдавший тест(ы) на неудовлетворительную оценку, должен их пересдать.

Примерный перечень практических заданий для практических занятий и лабораторных работ

1. Ознакомьтесь со справкой по каждой из команд.
2. Определите текущую дату и время.
3. Определите день недели, в который вы родились.
4. Выведите на экран список всех каталогов и файлов каталога /usr.
5. Выведите на экран список всех каталогов и файлов каталога /usr/local.
6. Выведите на экран список всех каталогов и файлов, имена которых содержат три символа из каталога /usr/share.

7. Выведите на экран список всех каталогов и файлов, имена которых начинаются на 'd' из каталога /usr/ bin .
8. Создайте систему каталогов и файлов согласно схеме.



9. Скопируйте файл Feihtvanger.1 в каталог usr/library/history.
10. Скопируйте файл Druon.2 в каталог usr/library/history.
11. Удалите файл Ahmatova.2 из каталога usr/library/liric/russian.
12. Выведите на экран содержимое каталогов usr / library/history и usr / library/history /german.
13. Организуйте конкатенацию (слияние) файлов LTolstoy .1 и ATolstoy .2 в каталоге usr / library / classic / russian .
14. Удалите каталог classic.
15. Выведите на экран список ранее выполненных команд и покажите его преподавателю.
16. Выйдите из системы.

Лабораторные работы (примерные задания)

17. Написание программы с использованием getpid() и getppid()
В качестве примера использования системных вызовов getpid() и getppid() самостоятельно напишите программу, печатающую значения PID и PPID для текущего процесса. Запустите ее несколько раз подряд. Посмотрите, как меняется идентификатор текущего процесса. Объясните наблюдаемые изменения.

18. Прогон программы с fork() с одинаковой работой родителя и ребенка

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int main()
{
    pid_t pid, ppid;
    int a = 0;
    (void)fork();

    /* При успешном создании нового процесса
       с этого места псевдопараллельно
       начинают работать два процесса: старый
       и новый */
    /* Перед выполнением следующего выражения
       значение переменной a в обоих процессах
       равно 0 */

    a = a+1;

    /* Узнаем идентификаторы текущего и роди-
```

```
       тельского процесса (в каждом из
       процессов !!!) */

    pid = getpid();
    ppid = getppid();

    /* Печатаем значения PID, PPID и вычислен-
       ное значение переменной a (в каждом из
       процессов !!!) */
    printf("My pid = %d, my ppid = %d,
      result = %d\n", (int)pid, (int)ppid, a);
    return 0;
}
```

Наберите эту программу, откомпилируйте ее и запустите на исполнение (лучше всего это делать не из оболочки mc, так как она не очень корректно сбрасывает буферы ввода-вывода). Проанализируйте полученный результат.

19. Написание, компиляция и запуск программы с использованием вызова fork() с разным поведением процессов ребенка и родителя

Измените предыдущую программу с fork() так, чтобы родитель и ребенок совершали разные действия (какие – не важно).

20. Написание, компиляция и запуск программы, распечатывающей аргументы командной строки и параметры среды

В качестве примера самостоятельно напишите программу, распечатывающую значения аргументов командной строки и параметров окружающей среды для текущего процесса.

21. Прогон программы с использованием системного вызова exec()

Для иллюстрации использования системного вызова exec() давайте рассмотрим следующую программу

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int main(int argc, char *argv[],
         char *envp[]){

/* Мы будем запускать команду cat с аргументом
   командной строки 03-2.c без изменения
   параметров среды, т.е. фактически выполнять
   команду "cat 03-2.c", которая должна выдать
   содержимое данного файла на экран. Для
   функции execle в качестве имени программы
   мы указываем ее полное имя с путем от
   корневой директории –/bin/cat.

   Первое слово в командной строке у нас
   должно совпадать с именем запускаемой
   программы. Второе слово в командной строке
   - это имя файла, содержимое которого мы
   хотим распечатать. */

(void) execle("/bin/cat", "/bin/cat",
              "03-2.c", 0, envp);
```

```
/* Сюда попадаем только при
   возникновении ошибки */
printf("Error on program start\n");
exit(-1);
return 0;     /* Никогда не выполняется, нужен
                 для того, чтобы компилятор не
                 выдавал warning */
}
```

Откомпилируйте ее и запустите на исполнение. Поскольку при нормальной работе будет распечатываться содержимое файла с именем 03-2.c, такой файл при запуске должен присутствовать в текущей директории (проще всего записать исходный текст программы под этим именем). Проанализируйте результат.

22. Написание, компиляция и запуск программы для изменения пользовательского контекста в порожденном процессе

Для закрепления полученных знаний модифицируйте программу, созданную при выполнении задания раздела "Написание, компиляция и запуск программы с использованием вызова fork() с разным поведением процессов ребенка и родителя" так, чтобы порожденный процесс запускал на исполнение новую (любую) программу.

С остальными заданиями можно ознакомиться на сайте math.pstgu.ru

Критерии оценивания лабораторных работ: по 5 баллов – работает/не работает

Автор: Буянов С.В. к.т.н., доцент кафедры «Информатика» ФИПМ

Одобрено на заседании кафедры Информатики от «31»мая 2019 года, протокол № 05-19.